Online assessment of dimensional numerical answers using STACK in science

Online assessment of dimensional numerical answers using STACK in science

C J Sangwin
School of Mathematics
University of Edinburgh
Edinburgh, UK, EH9 3FD
C.J.Sangwin@ed.ac.uk
ORCID: 0000-0002-3725-8625

M Harjula
Department of Mathematics and Systems Analysis
Aalto University, Helsinki
FI-00076, Finland
matti.harjula@aalto.fi
ORCID: 0000-0002-0622-5177

February 6, 2017

Abstract

In this article we report research into the computer representation and automatic assessment of expressions which are dimensional numerical quantities. We consider how machines represent and manipulate dimensional numerical data. Through action research we examine how students enter dimensional numerical data into a machine for online assessment, and examine the properties of such answers which practical teachers seek to establish. We consider the extent to which these properties can be established automatically, and report the outcomes of our action research: extensions to the STACK online assessment system which implement both syntactic and semantic layers to represent dimensional numerical quantities in a form suitable for online assessment.

Keywords: E-learning, online learning, automatic assessment, educational technology.

1 Introduction

We report research into the computer representation and automatic assessment of expressions which are dimensional numerical quantities. Dimensional numerical expressions occur in all areas of STEM (Science, technology, engineering and mathematics), but particularly often in physics. Various international standards (see [1] and [2]) specify how to represent dimensional numerical information. However our goal is to implement an automatic online assessment system where students' answers are not always correct. Correct answers are all alike; every incorrect answer is incorrect in its own way. For example, even assuming a well-formed syntactic expression (not a given), a student might express the correct number but in the wrong form or using different units. While deciding if figures are or are not significant is well-specified in a formal sense there are interesting edge cases and choices about what to do with them in an assessment context. We adopt the action research paradigm, and the outcomes will guide the development of functionality in an existing online computer aided assessment (CAA) system.

1.1 Research questions

This article seeks to address the following questions.

1. How do machines represent and manipulate dimensional numerical data?
2. How do students enter dimensional numerical data into a machine for online assessment?

3. What properties of dimensional numerical data do teachers seek to establish of students' answers? To what extent can these properties be established automatically?

This is a very specific aspect of online learning, but any system which accepts answers from students which are dimensional numerical quantities will rely on the results of this research. Adding support for scientific units in CAA provides a paradigmatic example illustrating all stages of the automation process.

- Human-computer interaction associated with the input of expressions, notation and syntax.
- The representation and automatic manipulation of mathematical knowledge.
- Automatically establishing objective properties, and providing useful outcomes for students.
- Pedagogy and the use of online assessments.

To automate a process we must understand it. Our previous experience indicates that teachers gain valuable teaching insights by considering such questions and the outcomes of similar research. The articulation of this understanding will be valuable to any teacher, e.g. with more nuanced ways of giving specific feedback to their students. Any scientist seeking to assess students' answers online will also have to consider our questions and so our results will be widely applicable beyond system designers.

1.2 Methodology

We have adopted an action research approach, that is seeking to solve an immediate problem through a reflective process of progressive improvement to a practical online assessment system. This has much in common with design research, see [3]. To address our research questions we have implemented an online assessment system, and used this with large groups of students.

2 Background to contemporary computer aided assessment

For over a quarter of a century online assessment of mathematics has accepted a mathematical expression from a student as an answer, sought to automatically establish relevant objective properties and provide feedback. This goes beyond relying on multiple choice and similar question types with their well-known difficulties (see [4]). For example, a teacher might ask for a polynomial expression and have the computer seek to establish algebraic equivalent with the correct answer and that it is written in a particular algebraic form, such as factored. The teacher can also establish if a student's expression is consistent with a particular mistake and provide specific feedback about potential improvements on future performance. Research such as [5] has suggested such feedback is likely to be effective.

The following features are now typical in many, if not most, mathematical systems.

- Random versions of a particular question are generated in a structured way using computer algebra systems (CAS). Reverse engineering from the answer also provides steps in a fully worked solution. Independently, a random selection from a question bank can normally also be made to create an activity (e.g. quiz) for an individual student.
- Students provide the final answer in the form of a mathematical expression, e.g. an equation, rather than responding to multiple choice questions. It is not yet typical to automatically assess a complete argument or proof.
- Objective mathematical properties of answers are automatically established, e.g. algebraic equivalence with a correct answer.

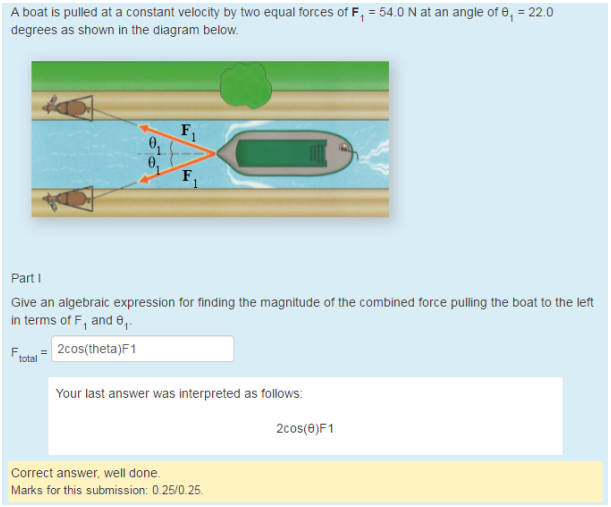

Figure 1: A school physics question in STACK

- Outcomes are automatically generated (including feedback) which fulfil the purposes of formative and summative assessment.
- Data on all attempts at one question, or by one student, are stored for later analysis.

Action research methodology needs a specific target implementation, but while this article describes a particular system there are many similar endeavours to create effective online assessments systems. [6] provided a comparison of those available in 2013. Inasmuch as mathematics is taught to physics these tools are available and potentially valuable. Science has additional subject requirements which we investigate here.

Our research uses STACK which was initially developed at the University of Birmingham from 2004. STACK has sustained development and use for over a decade with significant contributions of code from Aalto University Finland, the United Kingdom Open University and latterly the University of Edinburgh in Scotland. A demonstration server is available at `https://stack.maths.ed.ac.uk/demo` (retrieved November 2016). STACK was originally developed for Moodle but has been ported to ILIAS (see `http://www.ilias.de`, retrieved November 2016) and is used in other systems, including Blackboard, through the LTI protocol.

STACK has proved to be reliable at scale with thousands of users on over 600 registered Moodle sites. At the United Kingdom Open University during the academic year 2015-16, students attempted over 880,000 questions on seven modules. The STACK question type accounted for approximately 15% of all questions used, and is second only to multiple choice in popularity (at 35% of all questions). There are a number of international projects built around STACK such as the Abacus `https://abacus.aalto.fi/` material bank. STACK is used for diagnostic tests in Germany, [7] and for a major project in Portugal, [8]. Publishers are increasingly supplementing textbooks with online assessments. For example, Physics Curriculum & Instruction has chosen STACK as its solution for creating 1600 online homework problems to accompany its textbook *Physics Fundamentals* by Vincent Coletta, [9].

3 Review of scientific units in software

This section reviews support for units in current software from the perspective of online assessment.

Maple, (see `http://www.maplesoft.com/` retrieved November 2016), uses multiplication to attach

units to numbers. For example, the command line input is `90.4*Unit('km');` or `Unit(90.4*'km');` to attach units to a number. The Maple function `whattype` returns the operator of an expression, in this case $*$, indicating that normal multiplication is used to attach units to numbers. This approach has a serious drawback, as illustrated by the following Maple behaviour.

```
S:=0*Unit('s');
T:=0*Unit('m');
S+T;
```

returns 0. We would expect

```
Error, (in Units:-Standard:-+) the units `m` and `s`
have incompatible dimensions
```

In Maple unit names are separated from variable names so that it is still possible to use variables with the same name for other computations, which we believe is advantageous.

Mathematica (see `http://www.wolfram.com` retrieved November 2016) also has a comprehensive system for dealing with scientific units under the `Units` package. Mathematica input is very flexible, e.g. the `Quantity` allows natural-language for quantities with units.

Matlab (see `http://www.mathworks.com` retrieved November 2016), is primarily intended for numerical computing. The Simscape component of Matlab has a library of standard units, enabling a user to specify the units of a physical signal. The MuPAD add-in provides support for scientific units, attaching units to numbers with multiplication using syntax such as `27*unit::cm + 30*unit::mm`.

Most CAS support integer data types and at least one float implementation. Fixed precision floats are used for speed while arbitrary precision numeric types trade efficiency with absolute precision. Unfortunately arbitrary precision arithmetic can be limited to special functions and is not supported by all operations. This makes using it with student input expressions difficult.

More seriously for eduction, all decimal number representations lack the ability to carry metadata relating to the original input representation. In particular, representations do not keep track of the trailing zeros given during input and input such as `0.0100` is immediately represented as `0.01`. If a scientist has used the last two digits to signal three significant figures, important information has been lost. Floating point numbers also only aim to find the closest binary fraction presentation for the mantissa which may result in a different number from the original. Any new value most definitely has a different number of trailing zeros or significant digits. For example, Maxima converts decimal input `1.200` to the floating point 1.20000000476837 because one and a fifth cannot be represented exactly as a finite binary float. This process, well-described in [10], has serious implications for assessment: we need both a syntactic layer and a semantic layer to represent numbers but none of the existing software available to us has this functionality.

4 Results: developing support for scientific units

STACK uses the computer algebra system Maxima, `http://maxima.sourceforge.net/` (retrieved November 2016). Everything in Maxima is either an *atom* or an *expression* tree. Atoms are the most basic data types, e.g. integers, floats, strings, function names or variables. Non-atomic expression trees consist of an *operator* and *arguments* which represent the application of a function (the operator) to its arguments. The arguments are either atoms or expressions. For example in $1 + x^2$ we have the operator $+$ with its arguments 1 and x^2. The first of these is an atom (integer one), while the second is itself an expression x^2.

Even sets, such as $\{1, 2, a\}$, are represented by the operator "`set`" with three arguments. The interface accepts traditional notation `{1,2,a}`, which corresponds to users' expectations and `set(1,2,a)`. Trees provides a unified representation but some aspects might appear strange. The `set` function appears to do

nothing, however when simplifying a set duplicate elements are removed and remaining arguments are re-ordered. For assessment users need fine control over how all functions operate. Normally we perform addition of $1 + 2$ evaluating the sum to establish it is equivalent to 3. At other times $+$ remains inert to help distinguish the difference between the representations 3 and $1 + 2$. Teachers often need to select algebraic properties (commutativity and associativity) to automatically establish that $x + x + y$ is the "same" as $x + y + x$ but "different" from $2x + y$ in another sense. These subtle judgements are needed by question authors to encode online assessment. We have found that it is much more effective for question authors to think, and encode their questions, in terms of explicit mathematical properties such as "algebraic equivalence" rather than to write code such as `if simplify(sa-ta)=0 then ...` as a proxy for equivalence.

4.1 Input, notation and syntax

For assessment it is essential that the students' interface corresponds closely with their expectations and traditional notational conventions. In practice there is ambiguity, inconsistency (genuine, between subject areas and internationally) and syntactically mal-formed input.

A key design features of STACK is the separation of "validation" from establishing any properties relevant to the question (assessment). Validation includes a syntax check, but goes beyond this to create a context for the question. If a dimensional numerical quantity is expected STACK is able to reject a number as invalid with a message that units are expected. Validation significantly reduces the number of situations in which students feel they are being penalised on a technicality, and it significantly increases the robustness of the subsequent evaluation of the student's answer. This is essential where feedback on academic merit is only available after a deadline. The student knows what they are supposed to do, and the teacher can be confident they have done so. A teacher might want to test if a student knows they are supposed to use units, and always to reject a purely numerical answer as invalid would remove the ability to test this. Options are available to the teacher to fine-tune the input validation for each situation. Example feedback is shown in Figure 2 starting *"Your last answer was ..."*. Separate feedback *"Correct answer, well done."* is provided to indicate that this question has been assessed and the outcome of the assessment.

Typically software uses multiplication, so that $4.1\,\mathrm{m/s}$ is typed as `4.1*m/s` or `4.1*m*s^(-1)`. STACK has always been able to allow teachers to permit input such as `4.1m/s` in situations where there is no ambiguity about implied multiplication. We could accept input `4.1m/s` or `4.1 m/s` (note the space) and convert to a valid Maxima expression `4.1*m/s`. Multiplication would be displayed as $4.1 \cdot m/s$ or worse $4.1 \times m/s$ and units should be in Roman typeface with a small space, see [2, p. 24]. Attaching units to a number is *not* the same as algebraic multiplication.

We have developed a dedicated *input type* to support scientific units in which teachers can require scientific units and in which assumptions are made about letters used to represent units. One feature of the input type is a check for common incorrect synonyms. For example, if a student types in `3.2mols` then STACK will suggest that while `mols` is not a known unit `mol` is. The input type checks for units in a case sensitive way and with more than one option STACK suggests a list. For example, for input `mhz` STACK suggests `MHz` or `mHz`. If the student types in `mHz` but really meant `MHz` there is nothing realistically to be done, other than mark it as incorrect. Knowledge of what students actually do in practice requires action research and cyclic development.

Physics commonly uses subscripted variables and users have asked us to support input of expressions such as P_{min} and ω_0 using the underscore character, e.g. `P_min`. Note that "min" is an existing Maxima function which is being used without its arguments. Missing function arguments could be considered a misuse of syntax for which users might expect an error message. Actually, the difference between a function, e.g. $\sin$, and the value of a function at a point, $\sin(x)$, is a rather subtle one and users sometimes do need to represent the whole function.

Using juxtaposition in subscripts as labels, e.g. a_{1x}, causes more problematic input issues. Input $1x$ is normally interpreted as an implied multiplication, so do we interpret `a_1x` as $a_{1\times x} = a_x$ or as $a_1 \times x$?

The notation a_n normally refers to the nth element of a sequence $a_1, a_2, \cdots$, and two labels could indicate two dimensional subscripting. Separating arguments with unencapsulated comas, such as $U_{E,a}$, creates problematic binding powers and it is much better to use an alternative form, such as `U[E,a]`. This gives meaning to the subscripts in a way which the purely presentational approach does not. The disadvantage is that users are required to remember a new input syntax. Greek letters have to be typed using the English name, and we currently do not support unicode characters in input. This makes input such as `v_xi` ambiguous between v_{xi} or v_ξ.

Some of these issues are an artifact of one dimensional typed input [11] but many persist with a drag and drop editor, or when inferring meaning from handwriting, voice recognition or mobile devices, see [12] and [13]. We hope users (especially students) will not notice when we accept ambiguous expressions, and perpetuate ambiguity in notation, although we know from experience that they will certainly complain when they feel we have got the design wrong.

4.2 CAS representation of dimensional numerical quantities

A variety of systems of units are currently used internationally and there is an even wider variety historically, see [1]. The International System of Units (SI), see [14], comprises a coherent system of units of measurement built on the seven base units, length, mass, time, electric current, temperature, luminous intensity and amount of substance.

We have developed a data type to represent dimensional numerical quantities based on SI and dimensional analysis. In this we split numbers from any units and use a (mostly inert) function as part of a Maxima tree. For example an expression `4.1*m/s` is represented internally as `stackunits(4.1, m/s)`. This internal representation is not normally seen by users. Users enter expressions using multiplication (explicit or implied by a space or juxtaposition). This representation enables us to fine-tune the display and to reliably deal with awkward edge cases such as $0\,\mathrm{s} \neq 0\,\mathrm{m}$. If multiplication were used then at some point the CAS will evaluate both $0\,\mathrm{s}$ and $0\,\mathrm{m}$ to 0 rendering them indistinguishable and loosing the dimensional information.

This very tight binding of units to numbers enables authors ensure that it is clear to which unit symbol a numerical value belongs and which mathematical operation applies to the value of a quantity. For example, [2] recommends authors use $35\,\mathrm{cm} \times 48\,\mathrm{cm}$ and not $35 \times 48\,\mathrm{cm}$. This point of view is reinforced in the NIST guidelines.

> Equations between quantities are used in preference to equations between numerical values, and symbols representing numerical values are different from symbols representing the corresponding quantities. When a numerical-value equation is used, it is properly written and the corresponding quantity equation is given where possible. (See Sec. 7.11.) [2, p. 8]

Internally we have chosen to use only unit symbols (e.g. m) and not unit names (e.g. 'metre'). This has a number of advantages. (1) We do not risk mixing unit symbols and unit names in a single expression, (2) we do not have conflicting international spelling (e.g. 'meter' and 'metre') and (3) input is much shorter. There are also disadvantages to this approach, e.g. the atom m is, in context, now a unit and not a free variable. Fortunately, SI does not have any serious conflicts internally in the choice of unit symbols.

4.3 Establishing objective properties for assessment

Assessing whether a student's answer is correct requires us to establish a number of separate properties.

1. Is the numerical part written in the correct form? (E.g. number of significant figures.)
2. Is the numerical part sufficiently close to the true number sought?

3. Are the correct units provided?

It is easy to confound the first two of these properties since for a correct answer, they will be indistinguishable. A student might express a number to three (say) significant figures but they have expressed the wrong number. To help students we want to provide very specific feedback and for this it is necessary to consider each property separately.

All comparisons on a computer are made by matching data structures which represent the information. Therefore, to compare $\mathrm{m/s}$ with $\mathrm{km/h}$ we need to convert both to a canonical representation. We do this using a basic dimensional analysis to convert any dimensional numerical quantity to SI-base units. This makes use of units and standard *prefix* values such as $\mathrm{k} = 10^3$, $\mathrm{M} = 10^6$ etc., see [2, Table 5]. For example `ohm` is defined to be `kg*m^2/(s^3*A^2)` and MHz is converted to $10^6/\mathrm{s}$. The consistency and coherence of the SI system makes writing code to perform this conversion relatively simple. With this foundation we can provide tools which help a teacher establish the properties relevant to their particular question. Transforming dimensional expressions back to base SI units also provides a mechanism to generate conversion factors for direct conversions between units.

A teacher might require certain units or they may choose to condone the use of compatible units. For example, with velocity a teacher might require $\mathrm{m/s}$ rather than $\mathrm{km/h}$. In a "strict" case the student is expected to use particular units and anything else is rejected, although an internal note for statistical purposes records where they have given compatible units. Where a teacher will condone the use of compatible units both expressions are converted to the SI base before any further comparisons.

Numerical accuracy is established in a number of ways. In some situations a teacher will require the student's answer to be expressed to a particular number of significant figures. In others situations a numerical tolerance will be stipulated, either relative to the teacher's answer (e.g. within 10% of the teacher's answer) or absolute (e.g. within 0.1 of the teacher's answer).

The design of STACK enables a teacher to chose as many different tests as are relevant in the given situation. These tests are a core part of the design, enabling teachers to think in terms of objective properties rather than worry about how to establish them reliably. They can also test for incorrect answers known to arise in a given situation, and provide feedback to students. For example, a teacher is also likely to want to establish when the student's answer is incorrect by multiples of ten, indicating the wrong position of the decimal point.

Numerical marks are also awarded by teachers on the basis of test outcomes, enabling partial credit as appropriate. A teacher could award full marks with a "strict" test and part marks where the student has used compatible but different units. A teacher can separate marks for units from any marks for correct numerical answers. In mathematics, at least, colleagues are unlikely to agree on partial credit and so the award of marks is left entirely to the individual teacher.

4.4 Assessment of significant digits

A history of the evolution of the rules for dealing with significant figures was given by [15] and these are now subject to standards such as [16]. These rules are not exact and there is ambiguity in deciding, from a written number alone, how many significant digits are expressed. An online appendix[1] to [15] gives a number of examples.

> The significance of trailing zeros for numbers represented without use of a decimal point can only be identified from knowledge of the source of the value. For example, a modulus strength, stated as 140 000, may have as few as two or as many as six significant digits. [16]

The following cases illustrate the difficulties in inferring the number of significant digits from only the written form of a number.

[1] http://dx.doi.org/10.1119/1.4818368

1. 0.0010 has exactly 2 significant digits.
2. 100 has at least 1 and maybe even 3 significant digits.
3. 10.0 has exactly 3 significant digits.
4. 0 has 1 significant digit.
5. 0.00 has exactly 2 significant digits.
6. 0.01 has exactly 1 significant digit.

With trailing zeros it is not always possible to tell from the written expression precisely how many significant digits are present, which is problematic for automatic assessment. The standards recommend this ambiguity is eliminated using exponential notation. For example, $1.40 \times 10^5\,\mathrm{Pa}$ indicates that the modulus is reported to the nearest 0.01×10^5 or $1000\,\mathrm{Pa}$. When a teacher is being strict we need a test which will distinguish 100 from $1.00e2$. In other situations where a teacher has asked for three significant figures, we might choose to condone 100 but not 100.0. Some teachers might want to accept 100 as correct to two significant figures.

Current CAS do not enable us to annotate numeric representations with metadata such as the number of significant digits in the input form of the number. For assessment we must track this information at an upper level and then provide it to the evaluation logic as needed. We need to pre-parse the raw student's input to capture relevant representational details before it gets transformed into the internal CAS representation. We therefore parse the input and produce upper and lower bounds for the number of significant digits present in the input value. For example, $1230 * m => [4, 3]$ and $0.0010e3 => [2, 2]$. Using this data we can check that the number of digits matches the expected number of digits and that the value represented matches the expected value. Note that the expected value might have been evaluated at the CAS level without any regards to the precision needed by the student. When we require absolutely correct answers we need to ensure expected values are rounded to that accuracy using the same rules expected from the students.

4.5 Limitations

STACK is an open source project and we have created something which we believe is already very useful, but it is not perfect. All similar systems have limitations, including similar commercial systems. Open-source projects provide an opportunity for a frank community-led discussion of the educational and technical issues which commercial providers might be more reticent about, either for marketing reasons or because of commercial confidentiality.

STACK uses Maxima which, like all finite state machines, has limits on the accuracy of floating point numbers [10]. Science, particularly physics, tends to make use of very large and very small numbers in a way pure mathematics does not which can strain the limits of numerical representation. It is often better to use rational numbers: randomly generate and manipulate integers and only divide by appropriate powers of ten at the last moment when displaying floating point numbers.

We have developed a system which constructs bounds for the number of significant digits. This provides the means to test expressions but it leaves certain cases. To fully test the addition rules of significance arithmetic we would need more metadata. For addition of measurement values we would need the accuracy of each measurement and the knowledge of which numeric values are measurements. During assessment those details are known to question authors and so the correct number of significant digits is explicitly decided by the author instead of automatically inferring this from raw expressions.

Our current implementation provides a consistent mechanism to make the whole of SI available to teachers and students. This means that some symbols, such as s and m, now have meaning and in this context are not also available as free variables. We anticipate the need to support other contexts in which variable names match names used in SI, and we anticipate a need to allow teachers to specify units of their own. Creating an environment which allows teachers to specify custom contexts is much more complex and

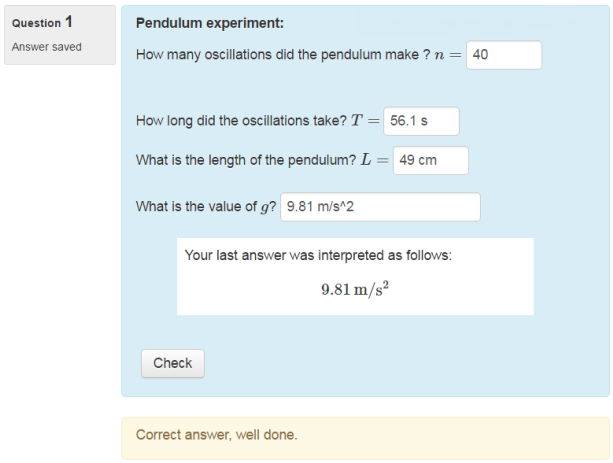

Figure 2: STACK question as part of a laboratory experiment

shifts the balance between simplicity of question authoring with reliable tools and writing each question from scratch. Further use in physics classes, using the existing features, will be needed to identify a sensible balance point between providing tools as part of the STACK distribution, and having question authors code from scratch in each question.

We have not, at this time, added support for United States customary units, although it has been requested and we are confident it could be done. While some colleagues in the United States are inching towards the metric system we do appreciate the need to support physics wherever it is taught.

5 Support for laboratory calculations with models

The CAS support in STACK makes it straightforward for a teacher to encode a precise model of a physical situation. Multi-part STACK questions can accept a student's experimental data and can establish if their subsequent calculation is consistent with accepted physical theory. Consider the classic simple pendulum experiment to measure the gravitational constant. A student measures the length of the pendulum and the time taken for an unspecified number of oscillations. The usual model predicts that for small oscillations the period is

$$T = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

where L is the length of the pendulum, g is gravitational acceleration, and T is the period. To calculate g we ask the student to enter (1) the length of their pendulum, (2) the number of oscillations counted, and (3) the time measured. They are expected to enter their calculated value for g and STACK also calculates this from their data. The teacher will need to encode what they think are reasonable constraints on the input data, and specify the required numerical accuracy of the resulting calculation.

Primarily STACK was designed to assess answers which are algebraic expressions, e.g. whether a student is able to rearrange the formula (1) to make g the subject. Where this is not done correctly the system could also use a student's incorrect formula with their data to establish if they have still correctly used their incorrect formula. Follow-through making of this kind is very easy to implement and STACK was designed with this in mind. However, in a context where immediate feedback is available we would question whether this is a desirable outcome. Students might better act on the immediate feedback to correct their formula and recalculate the value of g! In situations where no feedback is available such follow through marking is often considered to be a fair way to assess students.

Using automated assessment in an experimental setting goes back at least to the late 1980s where colleagues at the UK Open University have reported to us they were doing so with desktop computers. The flexibility of STACK, and the tools available through the underlying computer algebra system, make authoring questions much more straightforward than has been the case previously. Such assessments can readily be combined both with traditional laboratories or with virtual laboratory resources, see [17] and [18].

6 Conclusions

In this article we report research into the computer representation and automatic assessment of expressions which are dimensional numerical quantities. We have identified the need for both semantic and syntactic layers to represent dimensional numerical quantities. Through action research, and use with students, we have implemented data structures in the STACK online assessment system which represent and manipulate dimensional numerical quantities in ways which are appropriate for a wide range of assessment situations, including laboratory settings. Online assessment is here in many forms and is likely to stay. Commercial publishers are increasingly combining resources including traditional books, online assessments, video and other interactive media. There is also a vibrant open-source community of educators developing infrastructure and shared pools materials, e.g. projects like Abacus and the WebWork national problem library. These resources form an important component of our academic work, and are essential to the education of our students.

References

[1] F. Cardarelli. *Scientific Unit Conversion*. Springer-Verlag, New York, USA, 2nd edition, 1999.

[2] A. Thompson and B. N. Taylor. Guide for the use of the international system of units (SI). Special Publication 811, National Institute of Standards and Technology, Gaithersburg, MD 20899, March 2008.

[3] J. Van Den Akker, K. Gravemeijer, S. McKenney, and N. Nieveen, editors. *Educational Design Research*. Routledge, Oxford, 2006.

[4] C. J. Sangwin and I. Jones. Asymmetry in student achievement on multiple choice and constructed response items in reversible mathematics processes. *Educational Studies in Mathematics*, 2016.

[5] A. N. Kluger and A. DeNisi. Effects of feedback intervention on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, 119(2):254–284, 1996.

[6] C. J. Sangwin. *Computer Aided Assessment of Mathematics*. Oxford University Press, Oxford, United Kingdom, 2013.

[7] H. Barbas and T. Schramm. The Hamburg online math test MINTFIT for prospective students of STEM degree programmes. In *Proceedings of SEFI, Tampere, Finland*, 2016.

[8] R. C. Paiva, M. S. Ferreira, A. G. Mendes, and A. M. J. Eusébio. Interactive and multimedia contents associated with a system for computer-aided assessment. *Journal of Educational Computing Research*, 52(2):224–256, 2015.

[9] V. P. Coletta. *Physics Fundamentals*. Physics Curriculum and Instruction Inc., Lakeville, Minessota, 2nd edition, 2010.

[10] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *Computing Surveys*, 23(1):5–48, March 1991.

[11] C. J. Sangwin and P. Ramsden. Linear syntax for communicating elementary mathematics. *Journal of Symbolic Computation*, 42(9):902–934, 2007.

[12] Y. Nakamura, T. Taniguchi, K. Yoshitomi, S. Shirai, T. Fukui, and T. Nakahara. STACK project in Japan; item bank system, math input interface and question specification. In *Proceedings of the 13th International Congress on Mathematical Education*, 2016. TSG-44.

[13] Y. Nakamura and T. Takahara. Development of a math input interface with flick operation for mobile devices. In *12th International Conference on Mobile Learning, 9–11 April, Vilamoura, Algarve, Portugal*, 2016.

[14] The international system of units (SI). Technical Report 8th edition, Bureau International des Poids et Mesures, Paris, France, 2006.

[15] A. R. Carter. Evolution of the significant figure rules. *The Physics Teacher*, 51:340–343, September 2013.

[16] American Society for Testing and Materials, West Conshohocken, USA. *Standard Practice for Using Significant Digits in Test Data to Determine Conformance with Specifications*, e29-08 edition, 2008.

[17] P. A. Hatherly, S. E. Jordan, and A. Cayless. Interactive screen experiments: innovative virtual laboratories for distance learners. *European Journal of Physics*, 30(4):751–762, 2009.

[18] A. Potkonjak, M. Gardner, V. Callaghan, P. Mattila, C. Guetl, V. M. Petrović, and K. Jovanović. Virtual laboratories for education in science, technology, and engineering: A review. *Computers and Education*, 95:309–327, April 2016.